17/PRTS

09/830180
JC03 R PCT/PTO 23 APR 2001
REPLACED BY
ART 34 AMDT

A CIPHER

The present invention generally relates to a cipher and in particular to a cipher in which the secret cipher code which is required for both enciphering and deciphering is information which describes the process used to carry out enciphering.

Two commonly used types of cryptographic algorithms are private key algorithms which use a single shared key and public key algorithms which use two keys: a public key and a private key.

In these prior art algorithms the encryption process used is fixed although the particular encryption process can be selectable by user e.g. by using a particular encryption program (algorithm). The security of the encryption is provided by the key which is secretly exchanged between the encrypter operator and decrypter operator.

Such currently implemented ciphers are not easily scalable since they are defined for a specific block size and key size. In many instances the key is not big enough e.g. many ciphers have only 64 bit keys.

In accordance with a first aspect, the present invention provides an encipher apparatus for enciphering a signal that comprises a plurality of sequentially implemented cipher units, wherein each implemented cipher unit carries out an operation on the signals being ciphered which it can reverse. The apparatus also includes means for configuring the couplings between the implemented cipher units.

Another aspect of the present invention provides an encipher method wherein couplings between cipher modules are configured and the signal to be enciphered is sequentially passed through the configured cipher units. Each cipher unit is operative to carry out a process on the signal which it is capable of reversing.

Thus the present invention provides a universal cipher which is capable of implementing any cipher process. The encryption which is carried out on the signal is dependent upon the coupling configuration of the cipher units. The couplings are configurable without changing the configuration of the cipher units. This configuration is freely selectable and is preferably selected randomly or psuedo-randomly and automatically for the usual security reasons to prevent any element of predictability.

In one embodiment the cipher units are identical and thus perform identical reversible operations. The invention does however encompass the use of a plurality of types of cipher units wherein the sequential pattern of the different types is information that must be shared to allow deciphering of the signal encipher using the pattern.

The cipher units can be implemented in many different ways such as a reversible circuit either implemented in logic gates or in logic steps performed by a computer, analog circuitry, or optical elements. In fact, the encipher units can be implemented by any physical process which is reversible.

A further aspect of the present invention provides decipher apparatus for deciphering an enciphered signal and comprising a plurality of sequentially implemented cipher units wherein each cipher unit is implemented to carry out a process which it can reverse. The apparatus also includes means for configuring couplings between the cipher units dependent upon the enciphering of the enciphered signal wherein the couplings are configurable without changing the configuration of the cipher units.

Another aspect of the present invention provides a decipher method wherein couplings between a plurality of cipher units are configured and an enciphered signal is passed through the cipher units. Each of the cipher units carries out an operation on the enciphered signal which it is capable of reversing.

In one embodiment the cipher units are identical and carry out identical operations on the enciphered signal.

The use of reversible cipher units in both the encipher and decipher enable the configuration of the units to be the same although the implementation will be reversed. It is this reversibility which allows the use of information describing the cipher process to be used as a secret cipher code between the encipher and decipher. In other words, instead of a secret key to be shared by the sender and receiver of an encrypted message, a cipher code which describes the cipher process is shared instead.

Thus the invention is similar to the conventional symmetric cryptography technique except that there is no

single shared key but instead a single shared cipher code containing information describing the cipher process.

In a similar manner to use of a private key, the cipher code can be determined by either party in a two-party communication of an encrypted signal. In other words, either the recipient of an encrypted signal can request for the cipher code to be used and secretly pass this to the party for use in transmitting the encrypted signal, or the party transmitting the encrypted signal can secretly inform the recipient of the cipher code to be used to decrypt the signal.

The invention is equally applicable to the encryption of a signal which is not transmitted and which is instead stored securely e.g. the encryption and storage of data in a computer to prevent unauthorised access. In this example there need only be one party.

Because the cipher modules are reversible, the encipher and decipher can comprise the same apparatus. Thus, for duplex communication of an encrypted signal or for the storage of encrypted data for retrieval and decryption it is possible for the same cipher module to be used but in reverse order for deciphering.

In one embodiment each cipher unit comprises a controllable switch module having a plurality of inputs at least one of which acts to control a switching operation. Such a switch module can be implemented as a reversible gate such as a Fredkin's gate or an AND/NAND gate. Such gates can be implemented in logic either as

logic gates such as a programmable logic circuit or as logic steps implementing the gates in a computer program.

An advantage of this invention is that it is inherently scalable since the number of cipher modules can be varied dependent upon the configuration. Further, the size of the data block can be varied. This will also depend upon the configuration of the cipher unit.

Conveniently, ciphers are usually implemented digitally as a computer program. The programmability of a general purpose computer provides the facility for a universal cipher. Since a computer is capable of implementing a reversible computational process which implements a one-to-one mapping, and since general purpose computers are available which can be programmed to carry out any reversible computational process, any reversible computational process can be implemented thus implementing any one-to-one mapping. Any general purpose reversible computer can be used as a universal cipher. The use of a computer program to implement the reversible process further enables a user to select the type of reversible process to be implemented.

Since the present invention can be implemented on a general purpose computer by a suitable program, the present invention can be embodied as a storage medium storing instructions for controlling a processor e.g. a floppy disc, CD-ROM, smartcard, and programmable memory. Further, since the computer program can be transmitted over a network to be received and implemented on a computer, the present invention can be embodied as a

signal carrying the processor implementable instructions.

Embodiments of the present invention will now be described with reference to the accompanying drawings, in which:

Figure 1 is a schematic illustration of a cipher system.

Figure 2 is a schematic illustration of the reversibility of the cipher.

Figure 3 is a schematic illustration of the encipher unit.

Figure 4 is a schematic illustration of the decipher unit.

Figure 5 is a schematic illustration of a Fredkin's gate.

Figure 6 is a schematic illustration of the cipher code structure describing the configuration of Fredkin's gate.

Figure 7 is a schematic illustration of a gate circuit using Fredkin's gates.

Figure 8 is an illustration of the cipher code for the circuit of Figure 7.

Figure 9 is a functional diagram of a cipher code generator.

Figure 10 is a functional diagram of an encipher.

Figure 11 is a functional diagram of a decipher.

Figure 12 is a diagram of use of the cipher in the transmission of encrypted data.

Figure 13 is a diagram of a specific embodiment wherein the cipher code and possibly the cipher circuit is exchanged using a smartcard.

Figure 14 is a diagram of a processing apparatus capable of implementing the cipher.

Figure 15 is a flow diagram showing the generation and exchange of encrypted data using the cipher.

Figure 16a is a flow diagram illustrating the generation of the cipher code.

Figure 16b is a schematic diagram of the cipher code.

Figure 17 is a flow diagram illustrating the encipher process.

Figure 18 is a flow diagram illustrating the decipher process.

Figure 19 is a Fredkin's gate illustrated as a three-input logic gate.

Figure 20 is an implementation of the logic gate of Figure 19 using AND, OR and NOT gates.

Figure 21 is a diagram of an implementation of the Fredkin's gate using multiplexors.

Figure 22 is a diagram of an implementation of the Fredkin's gate using three-state buses.

Figure 23 is a diagram of an AND/NAND gate as an alternative reversible gate to the Fredkin's gate.

Referring now to the drawings, Figure 1 illustrates a cipher system in general wherein an encipher unit 10 generates an enciphered or encrypted signal using shared

configuration information. The configuration information is used to configure the encipher unit 10. The encrypted signal is then transmitted by a transmission medium 20 to a recipient decipher unit 30 which also has the shared configuration information. The decipher unit 30 is configured in accordance with the configuration information and operates the reverse of the process carried out by the encipher unit 10 to thereby decipher the signal.

Although in this embodiment a transmission medium 20 is illustrated, the transmission medium could simply comprise a storage medium on which the encrypted data is stored. Thus the operator generating the encrypted signal and the operator receiving the encrypted signal may in fact be the same.

Figure 2 schematically illustrates the reversibility of a cipher unit to act either as a decipher unit 30 or an encipher unit 10.

Figure 3 illustrates in more detail the cipher unit 10 which is comprised of cipher units 40a to 40d. Although in this embodiment four cipher units are illustrated, in a practical embodiment this would typically be at least four times the block size e.g. for a block size of 128 bits the number of cipher units is at least 512. The number will however depend on the level of security desired. As can be seen in Figure 3 the input signal is received at the input I of each of the cipher units 40a to 40d sequentially.

Figure 4 illustrates a decipher unit 30 in more detail. The decipher unit comprises the same configuration of cipher units 40a to 40d as in the encipher unit 10. However, in order to decipher the enciphered signal, it is passed in reverse through the cipher units 40a to 40d.

A specific implementation of the cipher will now be described in which the cipher unit is implemented using Fredkin's gates. Such a gate is illustrated in Figure 5. It is well known that a Fredkin's gate is both reversible (i.e. circuits implemented by Fredkin's gates can be run backwards to uncompute) and universal (i.e. can be used to design circuits that implement all one-to-one mappings).

In the Fredkin's gate as illustrated in Figure 5, the input A is used to control the exchange of data on inputs B and C. Thus Fredkin's gate performs a controlled exchange operation. If A=1, B and C are not exchanged i.e. B'=B and C'=C. If however, A=0, B'=C and C'=B. In mathematical notation $B'=AB+\bar{A}C$ and $C'=\bar{A}B+AC$.

Fredkin's gate is a convertive logic gate i.e. it preserves the numbers of 0's and 1's from the input to the output. In a cipher system this is undesirable, thus in order to break the conservation, NOT gates are selectively applied to the outputs to invert them. The selective inversion are operations which are inherently reversible and thus do not affect the reversibility of the circuit.

Having selected the type of reversible circuit used as the cipher unit, it is then necessary to determine a cipher code to describe the arrangement of the circuits. Figure 6 illustrates once such method wherein each Fredkin's gate is described by a four segment code. Each of the first three segments describe pin numbers to which the gates are attached in the input signal. The last segment is used to encode a description of the presence or the absence of inverters on each of the output pins A', B' and C'.

Consider an encipher process in which it is decided that the input signal is to be enciphered in 8 bit blocks. The input data thus comprises a 8 bit array indexed from 000 to 111. Each segment of the cipher code for each gate thus comprises a 3 bit code. For example the sequence 010 111 110 110 defines a gate with A of the gate attached to pin 2 (010) of the input, B attached to pin 7 (111) of the input and C attached to pin 6 (110) of the input. The last segment defines that output A' and B' are passed through NOT gates i.e. inverted. The 8 bit signal as modified by the first gate is then used as an input to the second gate and so on.

Figure 7 illustrates schematically an arrangement of 10 cipher circuits comprised of Fredkin's gates and NOT gates and Figure 8 illustrates the cipher code used to describe the circuit.

As can clearly be seen the cipher code simply comprises a digital code. The digital code is defined in as $(3 \times \log_2 N)+3$ bit blocks and each block defines a

cipher unit where N is the number of input bits i.e. the block size. The total number of bits to define a circuit is $M((3x\log_2 N)+3)$ where M is the number of cipher units. Whilst it is possible to allow a user to select a code freely by for example choosing a "password" in ASCII code which can be translated to binary (e.g. for the 8 bit input, 10 gate example in Figure 7, a 15 character 8 bit ASCII password could be used to describe the circuit), it is preferable for the usual security reasons to randomly generate a code which describes a random configuration of the gates.

In this example, in order to encrypt the signal it is passed from left to right through the gates. In order to decrypt the signal it is passed from right to left. Thus in decryption as the signal is input into each Fredkin's gate the mask defines whether it is to be first inverted before being switched in accordance with the value of the input on pin A.

It is possible for some of the gates to be implemented in parallel so long as their outputs and inputs are not coincident.

Figure 9 is a functional diagram of a cipher code generating apparatus. A random number generator 100 generates a random number to be used to form the cipher code. This is input through a validity checker 110 which checks whether the random number is valid. For example, the random number cannot result in two pins or more of the gate being on the same input line i.e. it defines a

valid gate. The validity checker 30 requires information on the block size in order to do this check.

The random number is then input into the cipher code (circuit array) forming unit 120 in order to build the cipher code describing the circuit array. The cipher code forming unit 120 also receives an input from the cipher code parameter selector 130 which is operable by a user to select the number of bits or cipher units to be implemented in the cipher, and to select the data block size. Also, in a general purpose computer, it is possible to select the type of reversible gates to be used. Since there is however a limited number of possible types of gates currently known which can be implemented reversibly, allowing such a selection does not greatly increase the level of security at present.

Once the cipher code has been formed it is then stored in a non-volatile memory 140 for use in enciphering and deciphering data. If data is to be transmitted between two parties, the cipher code must be secretly shared. Where there has been selection of parameters in building the cipher code i.e. the number of gates, the block size and the type of gates, this information will also need to be shared so that the cipher code can be used properly to implement a cipher circuit for both encryption and decryption.

Figure 10 is a functional diagram of an encipher apparatus. A signal to be enciphered is input by the data input device 200. This is then passed to a data block former 210 which forms the input signal into blocks

of data which can be sequentially passed through the encipher apparatus. The block of data is then passed into the working memory 220 as an array of N bits where N is the block size. A circuit implementor 250 then implements the cipher circuit in accordance with the circuit array stored in the non-volatile memory 260. The circuit array comprises a M array of gate descriptions, where each gate description comprises four segments (as illustrated in Figure 6). The circuit implementor 250 will operate on the data block in the working memory 220 to implement each of the gates sequentially. Circuit implementor 250 will therefore control the data block former 410 to input a block of data into the working memory 220 when it is ready to operate on it. Once the enciphering operation has been completed on the data block in the working memory 220, the circuit implementor 250 controls the passage of the data block from the working memory 220 into a memory 230. The encipher data block can then be passed out block-by-block into a data output device 240 which can either output the encipher data block block-by-block or can wait until all of the data blocks enciphered and output the complete enciphered signal.

Figure 11 is a functional diagram of the decipher apparatus in accordance with an embodiment of the present invention. Enciphered data is received by the encipher data input device 300 and is formed into enciphered data blocks by the enciphered data block former 310. The passage of enciphered data blocks into a working memory

320 is then controlled by a reverse circuit implementor 350. When a data block is in the working memory 320 the reverse circuit implementor 350 implements the encipher circuit in accordance with the circuit array stored in the non-volatile memory 260 in reverse. Once all of the gates of the circuit array have been implemented in reverse and thus the enciphered data block has been deciphered, it is output into the memory 330 under the control of the reverse circuit implementor 350. The decipher data block can then be output to the output device 340 which can then either output each of the data blocks sequentially or wait until the complete signal has been deciphered before outputting it.

Figure 12 illustrates an application of the cipher for the communication of enciphered data between computers 50, 51 and 52.

Computer 50 implements the encipher and generates enciphered data. This can either be stored on a non-volatile memory device such as a floppy disc 54 and passed to another computer 51 for deciphering, or it can be broadcast or transmitted over a network 53 for reception by computer 52 for deciphering there. Before the exchange of enciphered data however, it is necessary for the operators of computers 50 and 51 or 50 and 52 to secretly exchange the cipher code. This can be done by any conventional secret means such as a secure telephone call, a secure facsimile transmission or by letter, by courier or even by a secure e-mail.

Figure 13 illustrates another embodiment of the present invention wherein a computer 60 is provided with a smartcard programmer/reader 61. In this embodiment it is possible for a smartcard to be loaded with the decipher program as well as the cipher code. The smartcard can then be given to the intended recipient of enciphered data. Thus the intended recipient of the enciphered data can simply insert the smartcard into a smartcard reader and the processor on the smartcard will implement the deciphering circuit and thus inherently the enciphering circuit. Thus the smartcard can be used for both transmitting and receiving enciphered data. This embodiment can be used by an institution such as a financial institution (Alice). A user (Bob) will be issued with a smartcard and will be able to communication securely with the institution by inserting the smartcard into the reader 61 e.g. an automatic teller machine (ATM).

Figure 14 is a schematic diagram of the implementation of the cipher in a general purpose computer. The computer is provided with a bus 79 to communicate between operational units. A modem 70 is provided for connection over a telecommunications line 78 to transmit and receive enciphered data. Also a network card 80 is provided for connection over a network to transmit and receive data. A keyboard 74 is provided for inputting data and a display 71 is provided for displaying deciphered data. A processor 72 implements the encipher circuit either in a forward direction for

enciphering or in a reverse direction for deciphering in accordance with the circuit array stored in the memory section 77. The processor 72 operates in accordance with the circuit emulation program stored in the program memory 75. During the operation of the processor 72 data is temporarily stored in the working memory 76 and at the end of the enciphering or deciphering process the enciphered or deciphered data can be stored in the data storage device 73 which can comprise non-volatile storage media such as a floppy disc, a hard disc, a writable CD-ROM, or EPROM.

The method of operation of the cipher of this embodiment of the present invention will now be described with reference to Figures 15 to 18.

Figure 15 illustrates the steps involved in generation of the cipher code, enciphering of data, the transmission of the enciphered data and the deciphering of the enciphered data.

In step S1 a type of reversible processing is predetermined or selected e.g. Fredkin's gates. In step S2 the number of gates M and the block size N of the data is selected. In step S3 the cipher code (or circuit array) is then generated. In step S4 the cipher code is exchange secretly between Alice and Bob. In step S5 Alice enciphers data using a circuit configured according to the circuit array. Alice then communicates the enciphered data to Bob in step S6. In step S7 Bob deciphers the ciphered data using a reverse circuit configured according to the circuit array (cipher code).

Figure 16a illustrates in more detail the steps involved in a generation of the cipher code.

In step S10 a variable m is set to 0. This variable acts as the gate number. In step S11 a random number having P bits is then generated, where P is the number of bits necessary to describe a gate. In the example given hereinabove using Fredkin's gates, P=12 (4 segments each of 3 bits). In step S12 a check is carried out to determine whether this is a valid random number. One of the tests is whether the random number defines the gate having two or more pins on the same input data address which is not allowed. In step S13 if the random number is valid the gate number m is incremented and in step S14 the generated random number is stored indexed by m. In step S15 it is then determined whether random numbers have been generated for all of the gates i.e. m=M. If not, the process returns to step S11 for the generation of further random numbers. If random numbers have been generated defining all of the gates then the process ends in step S16.

Figure 16b illustrates the circuit array which comprises a PxM matrix. The matrix is indexed by M where each entry comprises P bits divided into four segments A, B, C and M each of 3 bits.

The process of enciphering data will now be described with reference to Figure 17.

In step S20 the data to be enciphered is input and the prestored circuit array (secret cipher code) indexed by m is read. The first N bits of data are then read as

a data block. If there are less than N bits of data, padding data is generated in order to make up N bits. The N bits of data are then loaded into the working array in step S22 and in step S23 the gate counter m is set to 1. In step S24 the first segment A for gate m in the circuit array is read and this is used to address a data bit from the working array in step S25. In step S26 it is then determined whether the read data bit is 0. If it is not 0 then there is no exchange of data between input B and C and the process proceeds to step S30. If it is 0 then data bits B and C are exchanged. Thus in step S27 the second and third segments B and C for the gate m in the circuit array is then read and these are used to address two data bits in the working array. In step S29 these data bits are then exchanged and the process proceeds to step S30 where a mask bit counter b is set to 1 to index the first mask bit for segment A.

In step S31 the b^{th} bit of the mask is read and in step S32 it is determined whether this is zero. If it is not zero the data bit in the working array addressed by the b^{th} segment is inverted in step S33 otherwise no action is taken. In the next step S34 is determined whether all of the mask bits have been read i.e. b=3 indicating that the mask bit for segment C has been read. If not the mask bit counter b is incremented in step S35 to index the next mask bit for segment B or C and the process returns to step S31. If all of the mask bits have been read it is then determined whether all of the gates have been implemented i.e. m=M in step S36. If not,

the gate counter m is incremented and the process returns to step S24. Otherwise in step S38 the working array is output as a block of enciphered data. In step S39 it is then determined whether the data has all been enciphered and if not, in step S40 the next N bits of data are input and padded if necessary and the process returns to step S22. Otherwise the process ends in step S41 since all of the data has been enciphered.

The process of deciphering enciphered data will now be described with reference to Figure 18.

In step S50 the enciphered data is input and the prestored circuit array indexed by m is read. The first N bits of enciphered data are then read in step S51. The N bits of enciphered data are then loaded into the working array in step S52 and in step S53 the gate counter m is set equal to M i.e. the first gate to be implemented is in fact the last gate in the array so that the gates are implemented sequentially in reverse. In step S54 the mask bit counter b is then set to the first mask bit for segment A and in step S55 the b^{th} bit of the mask is read. It is then checked in step S56 whether this is zero and if not the data bit in the working array addressed by the b^{th} segment is inverted in step S57 otherwise no action is taken. The process then proceeds to step S58 wherein it is determined whether all of the mask bits have been read i.e. b=3. If not, in step S59 the mask bit counter b is incremented to index the next mask bit for segment B or C and the process returns to step S55. If all of the mask bits have been read for the

mask segment, in step S60 the first segment A for gate m in the circuit array is read. A data bit in the working array addressed by this first segment A is then read in step S61 and it is determined whether this is zero in step S62. If it is zero the second and third segment B and C for gate m in the circuit array are read in step S63 and in step S64 the data bits in the working array addressed by the second and third segments B and C are read. These are then exchanged in step S65 and the process proceeds to step S66. If in step S62 the data bit addressed by the first segment A is not zero the process proceeds to step S66. In step S66 it is determined whether the process has just been carried out for the first gate i.e. m=1 indicating that the deciphering of the current block has finished. If not in step S67 the gate counter is decremented and the process returns to step S54, and if so in step S68 the working array is output as a deciphered data block. In step S69 it is determined whether all of the gates have been deciphered and if not in step S70 the next N bits of deciphered data are read. The process then returns to step S52. If in step S69 it is determined that all of the data has been deciphered, in step S71 any data corresponding to padding data in the last block is ignored and the process ends in step S72.

In the above enciphering and deciphering embodiment each gate is implemented in software sequentially.

In the embodiment described hereinabove the reversible circuit is implemented by Fredkin's gates. The

Fredkin's gate can be viewed as a three-input, three-output logic gate as illustrated in Figure 19. This can be implemented using AND, OR and NOT logic gates (which are not reversible) as illustrated in Figure 20. Of course, since the logic gates can only conduct signals one-way in order for the circuit to be reversible, it must perform an operation which is symmetric i.e. if the output of the circuit is put back as an input, the original input will be obtained. This is because a Fredkin's gate is an inverse of itself.

Thus the circuit illustrated in Figure 20 can comprise one of the cipher units 40a to 40d illustrated in Figures 3 and 4.

Another implementation of the Fredkin's gate can be chosed using multiplexer as illustrated in Figure 21. each multiplexer 400 and 401 receives two input signals and one control signal. If the control signal is zero then the first input signal is passed. If the control signal is one the second input signal is passed.

The Fredkin's gate can also be implemented by three-state buses as illustrated in Figure 22.

All of the three circuits given hereinabove can either be implemented in software using a computer program which generates the circuits and simulates them or using electronic hardwired circuits. It is thus possible for Alice and Bob to be supplied with off the shelf programmable gate array chips and with the software that downloads the cipher circuit description onto the chip. Such software languages for circuit descriptions

can for example comprise Verilog-HDL. In order for Alice and Bob to establish the secret communications, the downloading of the cipher circuit description will only be done once. When the circuit is implemented using hardwired circuits, Alice and Bob can use one circuit for encryption and one circuit for decryption. It is however possible to use only one circuit by downloading the circuit description at the time when communications take place. For example, if Alice wishes to send an encrypted message she downloads the cipher code (circuit array) onto the chip. If she receives a message she can download the decryption circuit onto the chip.

In the embodiment given above, Fredkin's gate is implemented in logic. However, Fredkin's gate can be implemented in many different ways, for example, it is possible to implement the Fredkin's gate in optics. The device which can be used to implement the Fredkin's gate in optics is the Mach/Zehnder interferometer switch. Such a switch is disclosed in a paper by J. Dommelly *et al* entitled "A Gallium Arsenide Electro-optical Interferometer Modulator", (Proc. 7th Topical Meeting on Integrated and Guided Wave Optics, Kissimmee 1984), the content of which is hereby incorporated in full by reference.

Although in the above embodiments, the use of Fredkin's gate has been described, the cipher units of the present invention can be implemented in many different ways. For example, another form of reversible universal logic is the AND/NAND gate (which is also known

as Toffoli's gate). The operation of the AND/NAND gate can be given by:

$$\begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_{n-1} \\ x_n \end{pmatrix} \rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_{n-1} \\ x_n \oplus x_1 x_2 K x_{n-1} \end{pmatrix}$$

The AND/NAND gate is illustrated in Figure 23. In this gate the input on (n-1) of the n inputs act to switch the n^{th} input by virtue of an AND gate receiving the (n-1) inputs and acting on an XOR gate on the n^{th} input. Details on this particular type of gate are given in the paper by T. Toffoli entitled "Bicontinuous Extensions of Invertible Combinatorial Functions" (Mathematical Systems Theory, Vol. 14, pp. 13-23) the content of which is hereby incorporated by reference.

The AND/NAND gate can be implemented not just in logic as illustrated in Figure 23, but by any physical system.

In a system for implementing the cipher units of the present invention, any reversible computational system can be used and the present invention is not limited to the use of circuit implementation. For example, reversible cellular automata can be used as described in "Computation and Construction Universality of Reversible Cellular Automata" by T. Toffoli (J. Comput. Sys. Sci.,

Vol. 15, 1977, pp. 213-231), the content of which is hereby incorporated in full by reference, a reversible Turing machine as described in "Logical Reversibility of Computation" by C. Bennett (IBM J. Res. Dev. 6, 1973 pp. 525-532), the content of which is hereby incorporated in full by reference, quantum computing, for a "billiard ball", model of computation as described in "Conservative Logic" by E. Fredkin and T. Toffoli (International Journal of Theoretical Physics, Vol 21. nos. 3/4, 1982) the content of which is hereby incorporated in full by reference, for example.

In the embodiments described hereinabove, for security, a random number generator is used in order to randomly generate a circuit configuration. The random number generator is not essential to the present invention but does increase the level of security. Any of the standard strong real number generators available for crypto-software libraries can be used, or a true physical random source can be used. The generation of random or pseudo-random numbers is well known in the art.

It will be apparent to the skilled person in the art that the present invention can be implemented by providing Alice and Bob with a program that generates random circuits and simulates them. Generally in the software implementation the circuits will only be generated and simulated using the cipher code (circuit array) when the signal is input to be enciphered or deciphered.

If the cipher is to be implemented using programmable hardware, a manufacturer will provide Alice and Bob with a conventional programmable gate array and logic to set it up to run as a cipher.

Although the present invention has been described hereinabove with reference to specific embodiments, it will be apparent to the skilled person in the art that the present invention is not limited to the specific embodiments and modifications will be apparent to the skilled person in the art within the spirit and scope of the present invention.

CLAIMS:

1. Encipher apparatus for enciphering a signal, comprising:

a plurality of encipher functional modules sequentially coupled to operate sequentially on the signal, each encipher functional module having a plurality of inputs and a plurality of outputs and being operable to carry out a reversible operation on the signal input to said inputs; and

configuring means for configuring couplings between the plurality of outputs and inputs between said encipher functional modules.

2. Encipher apparatus according to claim 1, wherein said encipher functional modules are of a single type.

3. Encipher apparatus according to claim 1 or claim 2, wherein each said encipher functional module comprises a controllable switch module, and at least one of said inputs acts to control a switching operation.

4. Encipher apparatus according to any preceding claim wherein each of said encipher functional modules comprises a reversible gate.

5. Encipher apparatus according to claim 3, wherein said reversible gate comprises a Fredkin's gate or an AND/NAND gate.

6. Encipher apparatus according to any preceding claim wherein said configuring means is operative to configure the couplings between said encipher functional modules in accordance with information describing the configuration of the couplings between said encipher functional modules.

7. Encipher apparatus according to claim 6, including means for receiving said information.

8. Encipher apparatus according to claim 6, including means for generating said information.

9. Encipher apparatus according to claim 8, wherein the generating means includes a random or pseudo-random number generator and is operative to use random or pseudo-random numbers generated by said random or pseudo-random number generator to describe in code the configuration of the couplings between said encipher functional modules.

10. Encipher apparatus according to claim 9, wherein said generating means is operative to use a respective random or pseudo-random number generated by said random or pseudo-random number generator to describe in code the configuration of inputs and outputs of a respective said encipher functional module.

11. Encipher apparatus according to any preceding claim, wherein each said encipher functional module comprises a logic gate which does not conserve logic.

12. Encipher apparatus according to any preceding claim, wherein said plurality of encipher functional modules form a programmable circuit.

13. Encipher apparatus according to claim 12, wherein said plurality of encipher functional modules comprise a programmable logic gate array, and said configuring means comprises a programming means for programming said programmable logic gate array.

14. Encipher apparatus according to claim 12, wherein said encipher functional modules comprise analogue electronic modules.

15. Encipher apparatus according to any one of claims 1 to 11, wherein said signal is an optical signal and said encipher functional modules comprise optical components.

16. Encipher apparatus according to any one of claims 1 to 11, comprising a programmable computing apparatus, wherein said encipher functional modules comprise a computer code routine implemented on said programmable computing apparatus.

17. Encipher apparatus according to claim 16, wherein said encipher functional modules comprise a computer code routine repeatedly implemented dependent upon configuration information from said configuring means.

18. Encipher apparatus according to claim 16 or claim 17, wherein said computer code routine implements a logic gate.

19. Encipher apparatus according to any preceding claim including first selection means for selecting a type of encipher functional module to be used from amongst a plurality of possible types of encipher functional modules, wherein said configuring means is adapted to configure the encipher apparatus to use the selected type of encipher functional module.

20. Encipher apparatus according to any preceding claim, including second selection means for selecting the number of said encipher functional modules to be used, wherein said configuring means is adapted to configure the

encipher apparatus to use the selected number of encipher functional modules.

21. Encipher apparatus according to any preceding claim including third selection means for selecting the number of said inputs and said outputs for said encipher functional modules, wherein said configuring means is adapted to configure said encipher functional modules to have the selected number of inputs and outputs.

22. Encipher apparatus according to any preceding claim including splitting means for splitting the signal across the inputs of a first said encipher functional module in the sequence.

23. A method of enciphering a signal, the method comprising:

configuring couplings between a plurality of inputs and a plurality of outputs between a plurality of encipher functional modules coupled sequentially; and

passing said signal through said encipher functional modules;

wherein each encipher functional module carries out a reversible operation on said signal and said encipher functional modules act sequentially on the signal.

24. A method according to claim 23, wherein the encipher functional modules are of a single type.

25. A method according to claim 23 or claim 24, wherein each said encipher functional module comprises a controllable switch module, and at least one of the inputs controls the switching operation.

26. A method according to any one of claims 23 to 25, wherein said encipher functional modules each act as a reversible gate.

27. A method according to any one of claims 23 to 26, wherein the couplings between said encipher functional modules are configured in accordance with information describing the configuration of the couplings between said encipher functional modules.

28. A method according to claim 27, including receiving said information.

29. A method according to claim 27, including generating said information.

30. A method according to claim 27, including generating random or pseudo-random numbers and using the generated random or pseudo-random numbers to describe in code the configuration of the couplings between said encipher functional modules.

31. A method according to claim 30, wherein a respective generated random or pseudo-random number is used to described in code the configuration of inputs and outputs of a respective said encipher functional module.

32. A method according to any one of claims 29 to 31, wherein said encipher function modules perform logic operations on said signal.

33. A method according to claim 32, wherein the logic operations to not conserve logic.

34. A method according to any one of claims 23 to 33, wherein said encipher functional modules comprise a programmable logic gate array and the configuring step includes programming said programmable logic gate array.

35. A method according to any one of claims 23 to 33, implemented by computer code on a computing apparatus, wherein said encipher functional modules comprise a computer code routine implemented dependent upon configuration information.

36. A method according to claim 35, wherein the computer code routine is implemented repeatedly dependent upon the number of said encipher functional units to be implemented.

37. A method according to any one of claims 23 to 36, including selecting the type of encipher functional

module to be used from amongst a plurality of possible types of encipher functional modules.

38. A method according to any one of claims 23 to 37, including selecting the number of said encipher functional modules used.

39. A method according to any one of claims 23 to 38, including selecting the number of inputs and the number of outputs for said encipher functional modules.

40. A method according to any one of claims 23 to 39, including splitting the signal across the inputs of a first of said encipher functional modules in the sequence.

41. Decipher apparatus for deciphering an enciphered signal, comprising:

a plurality of decipher functional modules sequentially coupled to operate sequentially on the signal, each decipher functional module having a plurality of inputs and a plurality of outputs and being

operable to carry out a reversible operation on the enciphered signal; and

configuring means for configuring couplings between the plurality of outputs and inputs between said decipher functional modules dependent upon the enciphering of said enciphered signal.

42. Decipher apparatus to claim 41, wherein said decipher functional modules are of a single type.

43. Decipher apparatus according to claim 41 or claim 42, wherein said configuring means is operative to use information describing the configuration of the couplings between said decipher functional modules.

44. Decipher apparatus according to claim 43, wherein said information describing the configuration of the couplings between said decipher functional modules is equivalent to a reverse description of information describing the configuration of the couplings between encipher functional modules used to encipher the enciphered signal.

45. Decipher apparatus according to claim 43 or claim 44, including means for receiving said information.

46. Decipher apparatus according to claim 43 or claim 44, including means for generating said information.

47. Decipher apparatus according to claim 46, wherein the generating means includes a random or pseudo-random number generator and is operative to use random or pseudo-random numbers generated by said random or pseudo-random number generator to describe in code the configuration of the couplings between said decipher functional modules.

48. Decipher apparatus according to claim 47, wherein said generating means is operative to use a respective random or pseudo-random number generated by said random or pseudo-random number generator to describe in code the configuration of the inputs and outputs of a respective said decipher functional module.

49. Decipher apparatus according to any one of claims 41 to 48, wherein each said decipher functional module comprises a controllable switch module and at least one said input acts to control a switching operation.

50. Decipher apparatus according to any one of claims 41 to 49, wherein each said decipher functional modules comprises a reversible gate.

51. Decipher apparatus according to claim 50, wherein said reversible gate comprises a Fredkin's gate or an AND/NAND gate.

52. Decipher apparatus according to any one of claims 41 to 51, wherein each said decipher functional module comprises a logic gate which does not conserve logic.

53. Decipher apparatus according to any one of claims 41 to 52, wherein said plurality of decipher functional modules form a programmable circuit.

54. Decipher apparatus according to claim 53, wherein said plurality of decipher functional modules comprise a programmable logic gate array, and said configuring means comprises a programming means for programming said programmable logic gate array.

55. Decipher apparatus according to any one of claims 41 to 52, wherein said decipher functional modules comprise analogue-electronic modules.

56. Decipher apparatus according to any one of claims 41 to 52, wherein said signal is an optical signal and said decipher functional modules comprise optical components.

57. Decipher apparatus according to any one of claims 41 to 52, comprising a programmable computing apparatus, wherein said decipher functional modules comprise a computer code routine implemented on said programmable computing apparatus.

58. Decipher apparatus according to claim 57, wherein said decipher functional modules comprise a computer code routine repeatedly implemented dependent upon configuration information from said configuring means.

59. Decipher apparatus according to claim 58, wherein said computer code routine implements a logic gate.

60. Decipher apparatus according to any one of claims 41 to 59, wherein said configuring means is adapted to configure the type of decipher functional module dependent upon the type of encipher functional modules used in the enciphering of the enciphered signal.

61. Deciphering apparatus according to any one of claims 41 to 60, wherein said configuring means is adapted to configure the number of decipher functional modules dependent upon the number of encipher functional modules used in the enciphering of the enciphered signal.

62. Decipher apparatus according to any one of claims 41 to 61, wherein said configuring means is adapted to

configure the number of inputs and outputs of said decipher functional modules dependent upon the number of inputs and outputs of the encipher functional modules used in the enciphering of the enciphered signal.

63. A method of deciphering an enciphered signal, the method comprising:

configuring couplings between a plurality of inputs and outputs between a plurality of decipher functional modules coupled sequentially; and

passing said enciphered signal through said decipher functional modules;

wherein each said decipher functional module carries out an operation on said enciphered signal which is reversible by respective said decipher functional modules and said decipher functional modules act sequentially on said enciphered signal.

64. A method according to claim 63, wherein said decipher functional modules are of a single type.

65. A method according to claim 63 or claim 64, wherein the configuration is carried out using information describing the configuration of the couplings between said decipher functional modules.

66. A method according to claim 65, wherein said information describing the configuration of the couplings between said decipher functional modules is equivalent to a reverse description of information describing the configuration of the couplings between encipher functional modules used to encipher the enciphered signal.

67. A method according to any one of claims 63 to 65, including receiving said information.

68. A method according to any one of claims 63 to 65, including generating said information.

69. A method according to claim 68, including generating random or pseudo-random numbers and using the generated random or pseudo-random numbers to describe in code the

configuration of the couplings between said decipher functional modules.

70. A method according to claim 69, wherein a respective generated random or pseudo-random number is used to described in code the configuration of the inputs and outputs a respective said decipher functional module.

71. A method according to any one of claims 63 to 70, wherein said decipher functional module is a controllable switch module, and at least one of the inputs controls the switching operation.

72. A method according to any one of claims 63 to 71, wherein said decipher functional modules each act as a reversible gate.

73. A method according to any one of claims 63 to 72, wherein said decipher functional modules perform logic operations on the enciphered signal.

74. A method according to claim 73, wherein the logic operations do not conserve logic.

75. A method according to any one of claims 63 to 74, wherein said decipher functional modules comprise a programmable logic gate array and the configuring step includes programming said programmable logic gate array.

76. A method according to anyone of claims 63 to 74, implemented by computer code on a computing apparatus, wherein said decipher functional modules comprise a computer code routine implemented dependent upon configuration information.

77. A method according to claim 76, wherein the computer code routine is implemented repeatedly dependent upon the number of said decipher functional units to be implemented.

78. A method according to any one of claims 63 to 77, wherein the type of decipher functional module is

configured dependent upon the type of encipher functional module used in the enciphering of the enciphered signal.

79. A method according to any one of claims 63 to 78, wherein the number of decipher modules is configured dependent upon the number of encipher modules used in the enciphering of the enciphered signal.

80. A method according to any one of claims 63 to 79, wherein the number of inputs and outputs of the decipher functional modules is configured dependent upon the number of inputs and outputs of the encipher functional modules used in the enciphering of the enciphered signal.

81. Apparatus for generating a cipher code, comprising:

a random or pseudo-random number generator;

encoding means for encoding information describing the configuration of couplings between cipher functional modules for enciphering and/or deciphering a signal using random or pseudo-random numbers generated by said generator; and

output means for outputting the information for use in enciphering and/or deciphering a signal.

82. Apparatus according to claim 81, including first selection means for selecting at least one type of cipher functional module to be used from amongst a plurality of possible types of cipher functional modules.

83. Apparatus according to claim 81 or claim 82, including second selection means for selecting the number of said cipher functional modules to be used, said encoding means being adapted to include the selected number in the encoded information.

84. Apparatus according to any one of claims 81 to 83, including third selection means for selecting the number of inputs and outputs of said cipher functional modules, said encoding means being adapted to include the selected number in the encoded information.

85. A method of generating a cipher code, the method comprising:

generating random or pseudo-random numbers;

encoding information describing the configuration of couplings between cipher functional modules for enciphering and/or deciphering a signal using the generated random or pseudo-random numbers; and

outputting the information for use in enciphering and/or deciphering a signal.

86. A method according to claim 85, including selecting a type of cipher functional module to be used from amongst a plurality of possible types of cipher functional modules.

87. A method according to claim 85 or claim 86, including selecting the number of said cipher functional modules used, said encoding means being adapted to include the selected number in the encoded information.

88. A method according to any one of claims 85 to 87, including selecting the number of inputs and outputs of

said cipher functional modules, said encoding means being adapted to include the selected number in the encoded information.

89. Cipher apparatus comprising the encipher apparatus of any one of claims 1 to 22 and the decipher apparatus of any one of claims 41 to 62, wherein said encipher functional modules and said decipher functional modules comprise the same functional cipher modules but implement in opposite order.

90. A cipher method for enciphering and deciphering a signal comprising the encipher method of any one of claims 23 to 40 and the decipher method of any one of claims 63 to 80.

91. Processor implementable instructions for controlling a processor to carry out the method of any one of claims 23 to 40, 63 to 80, 85 to 89 or 90.

92. A carrier medium carrying the processor implementable instructions according to claim 91.

93. A storage medium storing logic to configure a programmable logic gate array to carry out the method of any one of claims 23 to 40, 63 to 80, 85 to 89 or 90.

ABSTRACT

A CIPHER

A cipher is disclosed for enciphering and deciphering a signal which comprises a plurality of sequentially coupled cipher units, each cipher unit being operable to carry out a reversible operation on the signal. The couplings between cipher units can be randomly configured using a cipher code. The cipher code can be secretly shared between the encipher and decipher. A signal which is enciphered using this technique is thus deciphered using a randomly selected cipher circuit as described by the cipher code.